CSE 102 10-29-21

Supplementary lecture 2

Sec 9.2 : Selection Problem

Defn

The i^{th} order statistic of an array $A[1 \cdots n]$, consisting of n distinct elements is the i^{th} smallest element in A.

equivalently : the unique element that is greater than exactly $(i-1)$ other elements $(1 \le i \le n)$.

equivalently : the unique element that is greater than or equal to exactly i elements.

$i = 1$: minimum

$i = n$: maximum

equivalently : sort $A[1 \ldots n]$.

the i^{th} order statistic is now in position i.

Selection Problem

Given $A[1 \ldots n]$, where all elements are distinct, determine the i^{th} order statistic. $(1 \le i \le n)$

One way to solve: sort the array, and return $A[i]$.

In general this takes time $\Omega(n \log n)$.

RandSelect() solves this in (average case) $\Theta(n)$.

Recall: RandPartition (A, p, r)

→ plits $A[p \cdots r]$ into subarrays s.t.

$$A[p \text{----} (q-1)] < A[q] < A[(q+1) \cdots r]$$

↓ return

RandSelect (A, p, r, i) $(p-e: 1 \le i \le r-p+1)$

```
1.  if p=r
2.      return A[p]
3.  else
4.      q = RandPartition(A, p, r)
5.      k = q-p+1  # length of A[p ···· q]
6.      if k=i
7.          return A[q]
8.      else if i<k
9.          return RandSelect(A, p, q-1, i)
10.     else # k<i
11.         return RandSelect(A, q+1, r, i-k)
```

Exercises

- write non-randomized version, and prove its correctness.
- write a recurrence for worst case runtime. (basic operation: comp. of array elements.)

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ T(n-1) + (n-1) & \text{if } n \geq 2 \end{cases}$$

- show exact soln is

$$T(n) = \frac{n(n-1)}{2}$$

hence $T(n) = \Theta(n^2)$

Let $t(n)$ be the average # of array comp. performed by RandSelect$(A, 1, n, i)$

Assume: each permutation of $A[1..n]$ is equally likely, hence return value of RandPartition() is equally likely to be any q: $1 \le q \le n$.

it appears that $t(n) = t(n, i)$, but we'll that there is NO dependence on i. ∎

$$t(n) = \frac{\sum_{q=1}^{n} \left((n-1) + \mathbb{P}(i<q) \cdot t(q-1) + \mathbb{P}(i>q)\, t(n-q) \right)}{n}$$

where $\mathbb{P}(i < q \le n) = \frac{n-i}{n}$

and $\mathbb{P}(1 \le q < i) = \frac{i-1}{n}$

are probabilities of recurring left and right respectively.

Thus

$$T(n) = (n-1) + \frac{1}{n}\sum_{q=1}^{n}\left(\left(\frac{n-i}{n}\right)T(q-1) + \left(\frac{i-1}{n}\right)T(n-q)\right)$$

$$= (n-1) + \frac{1}{n^2}\left[(n-i)\sum_{q=1}^{n-1} T(q) + (i-1)\sum_{q=1}^{n-1} T(q)\right]$$

$$= (n-1) + \frac{1}{n^2}(n-i+i-1)\cdot\sum_{q=1}^{n-1} T(q)$$

$$\boxed{T(n) = (n-1) + \left(\frac{n-1}{n^2}\right)\cdot\sum_{q=1}^{n-1} T(q)}$$

see midt review solutions #2

we show: $T(n) = O(n)$.

obviously $T(n) \geq n-1 = \Omega(n)$, hence

$$\boxed{T(n) = \Theta(n)} \leftarrow \text{asymp. soln.}$$

Sec. 4.2 Strassen's Algorithm

Problem: multiply two $n \times n$ matrices A and B:

$$C = A \cdot B$$

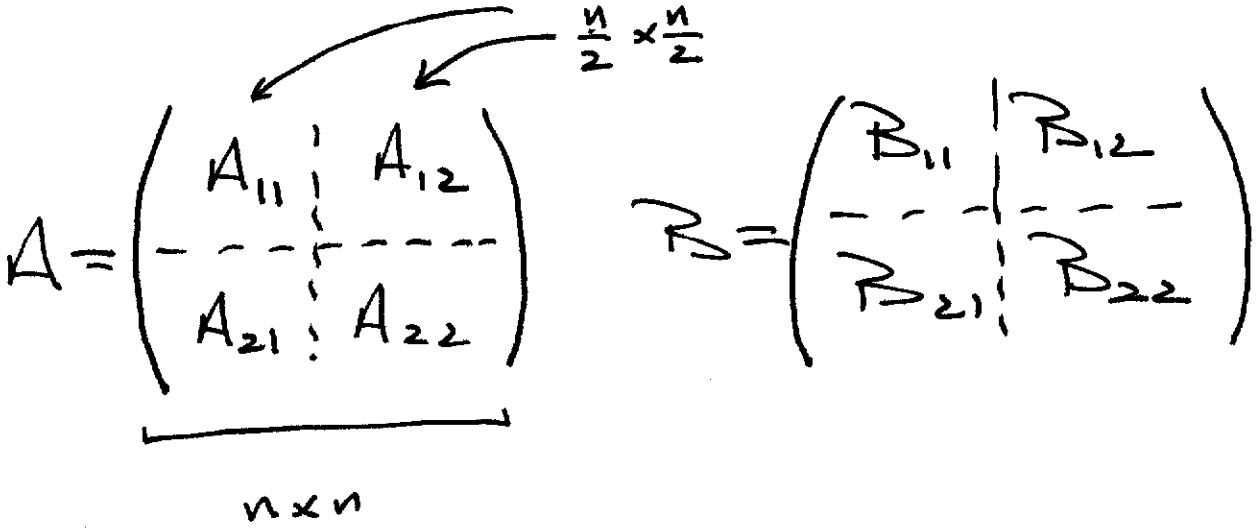
Recall, if $A = (a_{ij})$, $B = (b_{ij})$ then $C = (c_{ij})$ $(1 \le i \le n, 1 \le j \le n)$ where

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Iterative algorithm for this product has 3 nested loops of len n.
If we take basic op. to be multiplication of real no.s, then cost is $\Theta(n^3)$.

If n is an exact power of 2 ($n = 2^k$) we can do this recursively.

Partition A & B into 4 $\left(\frac{n}{2} \times \frac{n}{2}\right)$ submatrices:



Perform 8 multiplications of these submatrices.

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{12}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Then Combine

$$C = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right)$$

check this in A.R.

Exercise:

- write pseudo-code

Let $T(n)$ be # of basic OPS. Performed by this recursive algorithm. Observe

$$T(n) = \begin{cases} 1 & n = 1 \\ 8\,T\left(\frac{n}{2}\right) + 1 & n \geq 2 \end{cases}$$

↑ const. cost of dividing and combining. Book has $\Theta(n^2)$ here since they count additions.

Master thm:

Comp. 1 to $n^{\log_2(8)} = n^3$

↑ winner

By case 1: $\boxed{T(n) = \Theta(n^3)}$

↑ asmp. soln.

No better than iterative !!

In 1969 Volker Strassen discovered a method to do all this with only 7 multiplications of $(\frac{n}{2} \times \frac{n}{2})$ sub-matricies

See P. 80-81 of CLRS for details.

what is runtime ?

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 7T\left(\frac{n}{2}\right) + 1 & \text{if } n \geq 2 \end{cases}$$

Compare: 1 to $\underbrace{n^{\log_2(7)}}_{\text{winner}}$

by case 1: $\boxed{T(n) = \Theta\left(n^{\lg(7)}\right)}$ ← 2 asymp. soln.

note $n^{\lg(7)} = n^{2.8073\ldots} = o(n^3)$.

what if n is not an exact power of 2.?

Pad A and B extra rows & columns so their size is a power of 2.

Exercise: let N be the smallest power of 2, greater than or equal to n. define

$$A' = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right), \quad B' = \left(\begin{array}{c|c} B & 0 \\ \hline 0 & 0 \end{array}\right)$$

A': $N \times N$, B': $N \times n$

$$\therefore A' \cdot B' = \left(\begin{array}{c|c} A \cdot B & 0 \\ \hline 0 & 0 \end{array}\right)$$

Show that

$$N^{\lg(7)} = \Theta\left(n^{\lg(7)}\right)$$

so we don't lose anything in asymp. runtime.

Exercise

show how to do this for A, B not necessarily square.

say A, B are $n \times m$, define $N = ?$